Published online 12 May 2010 *Nucleic Acids Research, 2010, Vol. 38, Web Server issue* ***W595–W601***
doi:10.1093/nar/gkq398

GRAPE: GRaphical Abstracted Protein Explorer

Gregory Cipriano[1,*], Gary Wesenberg[2], Tom Grim[1], George N. Phillips Jr.[2] and Michael Gleicher[1]

[1]Department of Computer Sciences, University of Wisconsin-Madison Madison, WI 53715 and [2]Department of Biochemistry, University of Wisconsin-Madison, Madison, WI 53706, USA

Received February 24, 2010; Revised April 27, 2010; Accepted April 29, 2010

ABSTRACT

The region surrounding a protein, known as the surface of interaction or molecular surface, can provide valuable insight into its function. Unfortunately, due to the complexity of both their geometry and their surface fields, study of these surfaces can be slow and difficult and important features may be hard to identify. Here, we describe our GRaphical Abstracted Protein Explorer, or GRAPE, a web server that allows users to explore abstracted representations of proteins. These abstracted surfaces effectively reduce the level of detail of the surface of a macromolecule, using a specialized algorithm that removes small bumps and pockets, while preserving large-scale structural features. Scalar fields, such as electrostatic potential and hydropathy, are smoothed to further reduce visual complexity. This entirely new way of looking at proteins complements more traditional views of the molecular surface. GRAPE includes a thin 3D viewer that allows users to quickly flip back and forth between both views. Abstracted views provide a fast way to assess both a molecule's shape and its different surface field distributions. GRAPE is freely available at http://grape.uwbacter.org.

INTRODUCTION

One goal of structural biology is to understand the chemical and physical properties of macromolecules (especially proteins) and how this enables the chemical reactions behind life's processes. In order to study these large and complex molecules, researchers rely on visualizations that provide various levels of abstraction. The more abstract visualizations, such as ribbon diagrams, are limited to the portrayal of a molecule's internal structure. However, protein interactions involve the 'functional surface' presented: to a large degree, the internal structure simply exists as scaffolding to place various forces and chemical properties in proper spatial relationships with one another. This articles describes a web server that can generate and display abstract visualizations of this surface.

Popular molecular rendering programs, such as P_yMOL (1) and Chimera (2), build a visual representation of the functional surface of a protein by sampling fields, such as charge, onto a triangle mesh, resulting in an image such as in Figure 1a.

This triangle mesh can be thought of as the result of rolling a probe sphere over all the atoms in the protein. Connolly (3,4) provided practical methods for sampling these surfaces, which have subsequently been refined in both efficiency and quality (5–8). The resulting surface, called a 'solvent-excluded surface' is locally smooth, but at scales larger than an atom, it exhibits high-frequency detail. For even the smallest proteins, this detail can obscure significant surface features, such as pockets and clefts. Charge sampled on the surface often exhibits similar high-frequency detail, which can obscure significant patches of uniform charge, in addition to making the shape of the surface even harder to discern.

In Cipriano and Gleicher (9), we describe a method to overcome these limitations. Termed 'Molecular Surface Abstraction', the technique produces a simplified representation of both the geometry of, and of the surface fields surrounding, the molecular surface. Inspired, in part, by the works of David Goodsell and Olsen (10), this combination not only allows users to quickly see a gestalt of the surface, but also serves as a convenient canvas to place surface labels (or 'stickers') representing additional information, such as the location of external hydrogen bond donors and acceptors, or regions of known molecular interaction.

To date, however, software implementing surface abstractions has not been made widely available. While we have written a stand-alone application to test our ideas, the requirement that users download, install and then learn a completely new application before exploring abstractions, has inhibited their wider adoption.

*To whom correspondence should be addressed. Tel: 608 265 2711; Email: gregc@cs.wisc.edu

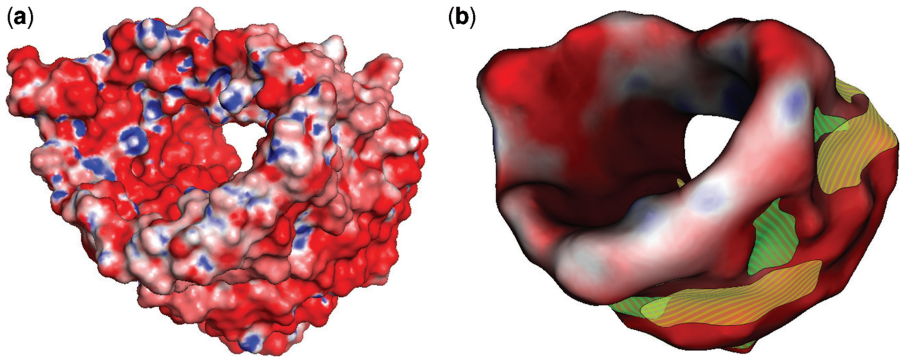

Figure 1. An example of a porine protein (PDB ID: 2POR), rendered (**a**) with PyMOL (1) and (**b**) after abstraction and stylization in GRAPE. Note how its large-scale shape and electrostatic potential distribution can be more readily seen after abstraction. High-quality lighting helps to further emphasize the ion channel, by providing important depth cues that are missing in the PyMOL rendering.

GRAPE represents the first completely web-based system for constructing and displaying abstracted molecular surfaces, using Protein Data Bank (PDB) data as input. Its functionality can be broken into two categories:

- A back-end system that takes PDB files as input, coordinates the job of abstraction among a server pool and produces an output file encapsulating the abstracted surface and its varied fields.
- A front-end viewer that uses a combination of Java and OpenGL to embed a live, explorable representation of the abstracted surface.

Project goals

GRAPE is motivated by the dual goals of making surface abstractions easy to generate and to give as many users as possible the ability to use them on their own molecules. We have organized GRAPE so that all computations, including those for high-quality lighting and mesh generation, are performed on the server; the viewer is a thin client that merely reads in the results of those computations and renders them. This reduces the computational burden on the user's computer and ensures that our system is available to users with low-end hardware.

In recent years, many online protein viewers have come to be widely used, such as Jmol (11). These viewers are ideal for presenting a low barrier to entry for exploration: no software needs to be installed and results are available from any computer regardless of platform. GRAPE is designed to be similarly accessible: the client software itself is quite small and requires only modest graphics hardware for texture rendering. For this version, because we do require graphics hardware, layering our functionality on top of existing molecular viewers was not feasible.

SERVER-SIDE PROCESSING

The process of creating and using an abstraction in GRAPE has three steps: first, obtain the data about a molecule; then, abstract it into a useful form; finally, load this data into a viewer. The first two steps are performed on the server and result in a compressed data file.

Input data

GRAPE takes as input a PDB file that may be either uploaded from a local copy or fetched directly from the PDB (12). Optionally, users may also upload a PQR file to supply a custom protonation state, overriding the automatic protonation computations done within GRAPE.

Abstracting a protein can take a long time, depending on its size and complexity, so the GRAPE server creates a separate job for each submitted protein. Jobs run asynchronously on the server; after a submission, users are redirected to a job queue to monitor the status of their job.

The job queue

Along with the PDB file itself, each job has the following metadata associated with it:

- JobID: the unique identifier for this job
- Tool: GRAPE
- Job Name: the name a user assigns to this job (not the name of the protein)
- Status: Active refers to a job that is currently running. LaunchView appears when a job has successfully finished. Clicking this will take you to the viewer. pqr/dx map failure indicates that either pdb2pqr or APBS failed to produce charge and SurfAbstract failure indicates that abstraction failed, for an unknown reason. For either of these failure modes, clicking the Job ID, and then clicking on the 'log' link on the following page, may indicate the source of the failure.

This data can be seen by browsing to the 'job queue' page, as shown in Figure 2.

Authentication

To allow users to better manage their queue, GRAPE provides an optional authentication mechanism. Users who wish to authenticate may create an account by first providing a username and password. They then gain the ability to filter the job queue to show only their jobs, to

Display my jobs only jobs for all users

Submitted within 24 hours 3 days 14 days 60 days 360 days anytime

Using Tool chipD GRAPE any Tool

Refresh Table Using New Settings

Job ID	Tool	Job Name	Status	Sumitted (CDT)	Username
1741	GRAPE	1egz	Active	2009-12-30 10:10:57	guest
1740	GRAPE	3EOJ	LaunchView	2009-12-30 10:03:30	guest
1739	GRAPE	2POR	LaunchView	2009-12-30 10:01:47	guest
1738	GRAPE	1fin	LaunchView	2009-12-29 19:18:47	guest
1737	GRAPE	1eqy test	SurfAbstract failure	2009-12-29 19:18:14	guest
1735	GRAPE	1m07 kajsdlkjs	LaunchView	2009-12-29 17:51:11	guest
1731	GRAPE	2i5s test	pqr/dx map failure Attempt LaunchView	2009-12-29 13:46:26	guest
1730	GRAPE	test same	LaunchView	2009-12-29 13:31:02	guest

Figure 2. The GRAPE job queue. The job queue is a list of pending and completed jobs, along with information on what user submitted for each job and the time of submission.

receive an email when their jobs finish and to mark specific jobs as 'private'.

By default, all jobs in GRAPE are marked as public, which means that all GRAPE users will be able to view the results of a job. For users with sensitive data, such as prepublication proteins, the optional ability to mark a job as 'private' ensures that only that user will see the results.

The authentication and queue management infrastructure used in GRAPE is derived from work done for the KFC server (13).

Processing a job

All major processing takes place on the server back end, where jobs are farmed out to a cluster of computers, in first-come first-served order. Each computer takes on the task of abstracting a single protein. This can be divided into two phases: the 'data collection phase', in which the shape and electrochemical properties of the original 'solvent-excluded' surface is computed, and the 'abstraction phase'.

All of the server-side code described below is based on the algorithms found in Cipriano, and Gleicher (9).

Data collection. The data collection phase breaks down as follows:

(1) Compute the solvent excluded surface using MSMS (6).
(2) Compute electrostatic potential using PDB2PQR (14) (or a user-supplied PQR file) and APBS (15).
(3) Compute hydrophobicity by finding, for each point, the hydropathic index of the closest amino acid in the protein.
(4) Predict potential hydrogen bond donor/receptor locations by locating atoms in the side chains of each amino acid that are both potential donors/receivers and also near the solvent-excluded surface.
(5) Find 'interface regions', areas of the surface which are within a radius (1.6 Å) of the van der Waals surface of another part of the molecular assembly (i.e. a different chain in the PDB file).
(6) Compute ambient occlusion lighting (16) that darkens interior regions of the molecule (Figure 2). High-quality global lighting has been shown to be useful as a shape cue, as recently demonstrated in QuteMol (17), but cannot be done in real time without high-end hardware. The GRAPE server precomputes lighting, reducing hardware requirements for our users.

Abstraction. After collecting surface data, the second phase of a server job is to abstract this data, transforming the detailed results of the first phase into a (visually) simpler form. This again can be broken down into a series of steps:

(1) Smooth fields, such as electrostatic potential, along the surface. This removes small features, while preserving larger regions of consistent value.
(2) Perform a series of mesh restructuring operations on the mesh geometry to first smooth and then

'sand off' small bumps and pockets. The resulting mesh is a smooth approximation of the original solvent-accessible surface, which we call an 'abstracted' surface.

(3) Apply stickers directly to the surface, as seen in Figure 4. Due to its smoothness, it is much easier to apply stickers to (i.e. to parameterize) the abstracted surface than the original solvent-excluded surface. We take advantage of this fact to overlay existing surface coloring with several types of stickers. In this way, multiple surface properties can be shown simultaneously.

Surface data format

The final results of the abstraction process are compressed into a single ZIP file that stores the information required for the client to draw both traditional and abstracted views of the protein. This ZIP container contains a number of smaller files, in three major categories:

- Surface geometry: each chain in the PDB file is given its own set of surface geometry, one for the solvent-excluded surface and (for chains that are not hetero-compounds) one for the abstracted surface. Every surface is contained in a binary .PLY file.
- Stickers: the picture for each sticker on the surface is stored in a compressed .png file. Note that some stickers, such as those for hydrogen bonds, are only stored once and then reused over the surface.
- Surface Fields: one XML file connects surface geometry with field data, identifies the location of each sticker on the surface and matches chains in the PDB with each set of .PLY files.

The ZIP file, which ranges in size from 200 KB to 12 MB, depending on the size of the protein, is stored on the server in a job-specific location.

CLIENT-SIDE VIEWER

After the GRAPE server has completely finished a job, its status changes to a link titled 'LaunchView'. Clicking this link brings up the results of all abstraction computations. We have built a GRAPE viewer, shown in Figure 3, which can be run directly within the output web page. Standard

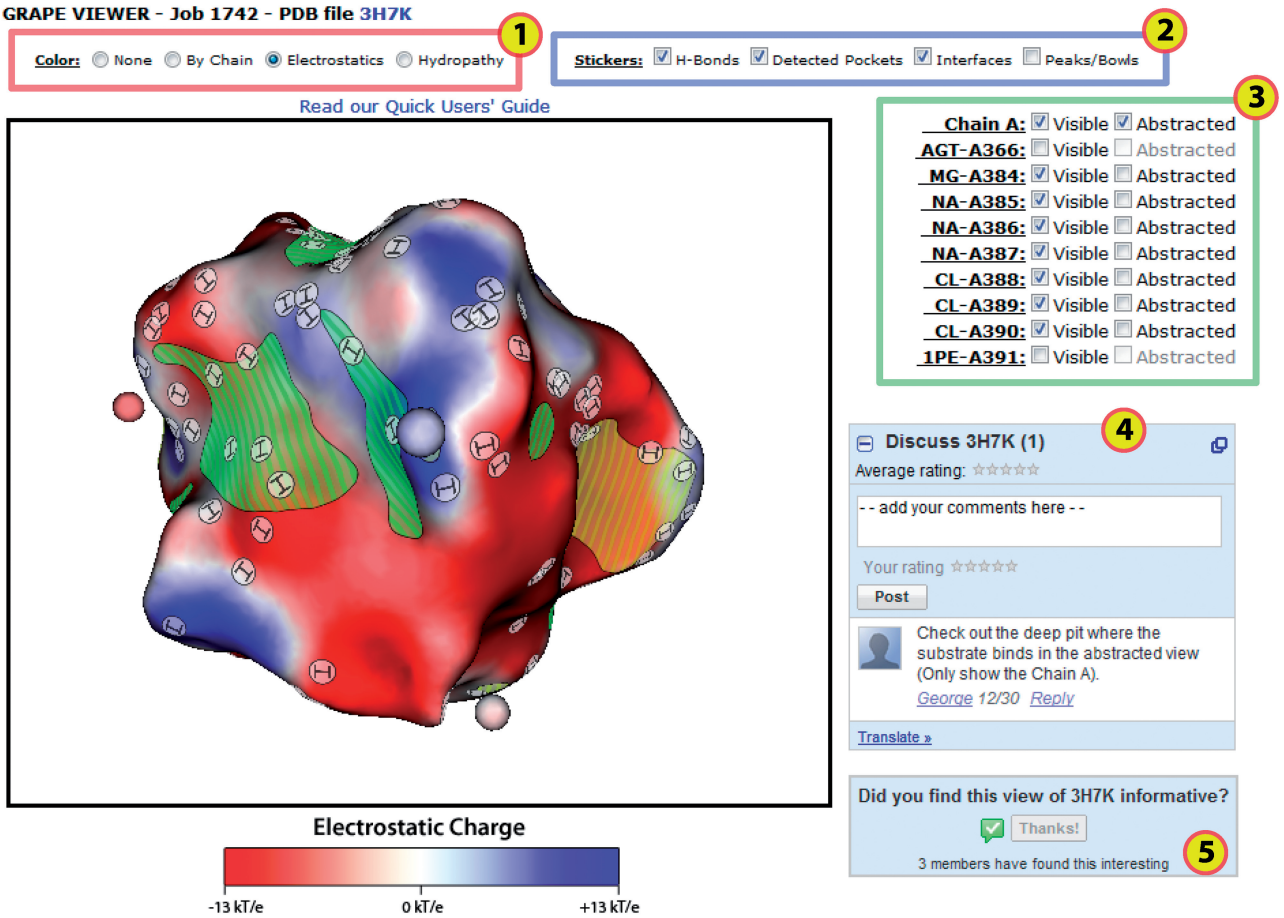

Figure 3. The GRAPE output window. Shown is the abstracted surface of a protein, each chain of which can be independently customized in the following ways: (1) Coloring: chains may be white, uniquely colored, colored by electrostatic potential or colored by hydropathic index; (2) the different sticker types may be hidden (Figure 4); (3) display: chains may be hidden, displayed as the solvent-excluded surface or displayed in abstracted form. Also on this page are: (4) the discussion gadget, where users may leave comments; and (5) the recommendation gadget, where users may recommend their favorite proteins to one another.

viewing controls are provided to let users navigate the molecular surface.

The viewer itself is written in Java, and uses the Java OpenGL (http://opengl.j3d.org/), or JOGL, binding library to render the surface. On page load, a small JAR file is downloaded for the GRAPE applet, followed by native JOGL libraries, if necessary. Finally, the ZIP file described in the 'Surface data format', section above is downloaded from the server and loaded into memory.

A link is also provided on this page to download abstraction data as a raw ZIP file. Currently, the GRAPE viewer is the only tool that can completely use this data, though we envision plugins for existing protein viewers that would allow us to merge abstract surfaces into existing methods of display.

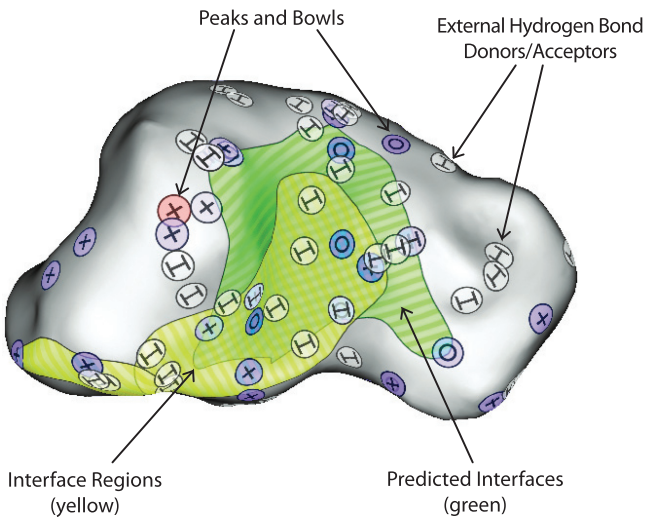

Figure 4. Stickers: four types of stickers are applied to abstracted surfaces, each of which may be independently turned on or off. 'Hydrogen bond stickers' are placed on the surface in areas that are close to one or more atoms that could form an external hydrogen bond. 'Detected Pocket stickers' indicate regions of the surface that resemble binding pockets, according to a variant of the LIGSITE pocket detector. 'Interface stickers' indicate regions of the surface in close proximity to another chain. And 'peak/bowl stickers' display as an 'X' or 'O', respectively, points where significant peaks or bumps in the original solvent excluded surface were removed.

SOCIAL NETWORKING

GRAPE uses Google Friend Connect throughout to foster discussion about protein surfaces, to link researchers together, to allow new users to quickly discover the surfaces that others have found interesting and to provide a mechanism for others to give us feedback about our tool.

Friend Connect gadgets expand the usefulness of the site in two ways: first, GRAPE uses the 'recommendation' gadget to give users the ability to recommend proteins to one another, as in Figure 6. So experienced users can discover interesting new models and new users can quickly see the benefits of abstraction on existing proteins, before they try their own. In addition to the these recommended proteins, we have added several of our own curated examples in a separate gadget, to ensure that new users always have a set of high-quality examples to begin their exploration of GRAPE.

Second, the viewer page for each job has an additional 'ratings and reviews' gadget, where users can discuss aspects of a single protein. An example can be seen in Figure 3.

DISCUSSION

In making abstractions more accessible for researchers to use, we made several trade-offs. As described in this 'Project goals' section above, our primary goal was to give researchers the ability to quickly use abstractions, to judge their utility for themselves. So rather than attempting to fit into the many different workflows used by researchers, we chose instead to build a simple, easy-to-use server that provides quick results. This decision does ultimately limit how useful our software can be: as it does not integrate into researchers'

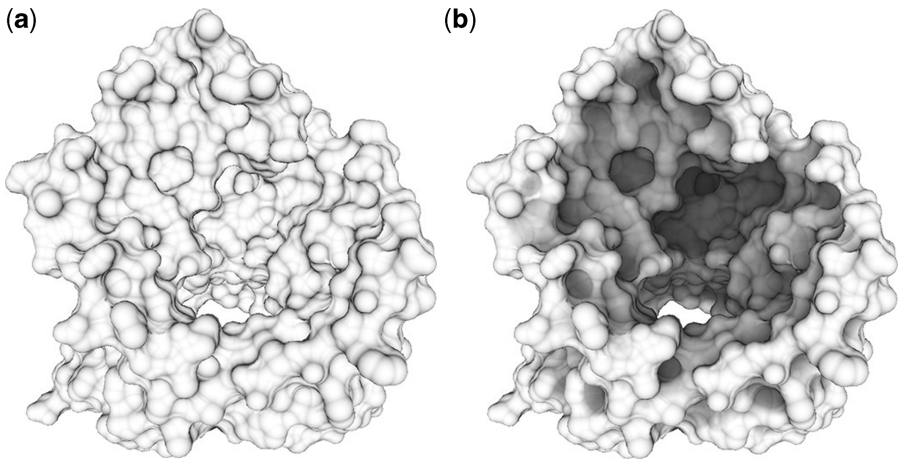

Figure 5. The same surface rendered (**a**) with and (**b**) without high-quality ambient occlusion lighting. Ambient occlusion lighting has been shown to be useful as a shape cue, as recently demonstrated in QuteMol (17), but can require high-end graphics hardware to compute. In GRAPE, this information is computed on the server and downloaded to the client for display.

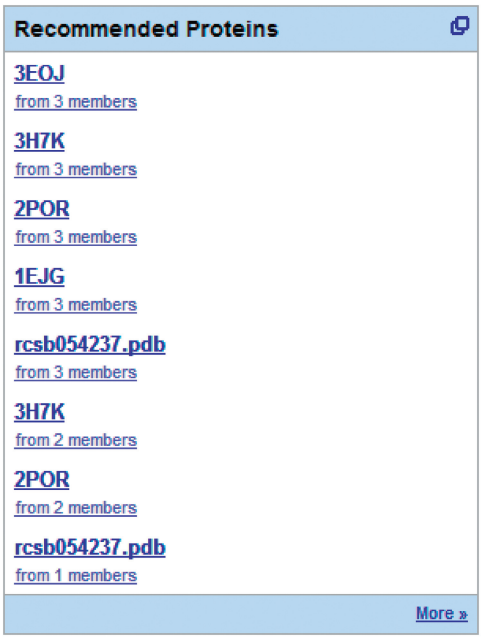

Figure 6. A GRAPE recommendation gadget. The proteins most recommended by GRAPE users are shown at the top of this gadget, which appears in several places on the site. Each link immediately directs a user to the viewing page for that protein.

workflows, they cannot use the tools to which they have grown accustomed. In the future, we would like to fix this limitation by providing a PyMOL plug-in that can understand and display our output data format.

We also chose to make the client hardware requirements as low as possible: the GRAPE viewer itself is very thin and all abstraction is done on the server. While a heavyweight client application could have provided more functionality, and potentially lowered the time between submitting a protein and viewing its abstraction, this would shift the burden to the client's computer, which would in turn limit the number of users who could use our software.

Nevertheless, the GRAPE viewer is currently missing several important features: there is no way to identify which regions of the surface are proximal to specific amino acids in the sequence, ribbon and stick-and-ball visualizations are not available, and certain parameters (such as the color maps for electrostatic potential and the degree of abstraction) are fixed. These features will be added in future revisions.

CONCLUSION

We have presented GRAPE, a web server that computes and displays abstracted views of the solvent-excluded surface of proteins. The server gives researchers a quick means to explore the surface of a protein of interest, in both abstracted and solvent-excluded views. In addition, GRAPE leverages social networking, in the form of Google Friend Connect, to foster discussion about any aspect of a protein, and to allow the community to share their most compelling, interesting and surprising proteins with one another.

ACKNOWLEDGEMENTS

We thank Julie Mitchell for providing much of the authentication and job management infrastructure and for encouraging our efforts to deliver abstractions as a web server. We also thank Joshua Oakgrove for designing and implementing an early version of the Java viewer and Aaron Bryden for helping us port server code to the Macintosh.

FUNDING

National Institutes of Health (NIH) training grant (NLM-5T15LM007359, in part); US Department of Energy Genomics:GTL and SciDAC Programs (DE-FG02-04ER25627, in part); National Science Foundation (NSF) awards (CMMI-0941013 and IIS-0946598, in part). Funding for open access charge: US Department of Energy (DE-FG02-04ER25627) funds.

Conflict of interest statement. None declared.

REFERENCES

1. DeLano,W. (2002) The PyMOL molecular graphics system, Schrödinger, LLC. http://www.pymol.org. (7 May 2010, date last accessed).
2. Pettersen,E.F., Goddard,T.D., Huang,C.C., Couch,G.S., Greenblatt,D.M., Meng,E.C. and Ferrin,T.E. (2004) UCSF Chimera - a visualization system for exploratory research and analysis. *J. Comput. Chem.*, **25**, 1605–1612.
3. Connolly,M. (1983) Solvent-accessible surfaces of proteins and nucleic acids. *Science*, **221**, 709–713.
4. Connolly,M.L. (1993) The molecular surface package. *J. Mol. Graph.*, **11**, 139–141.
5. Boissonnat,J.-D., Devillers,O., Duquesne,J. and Yvinec,M. (1994) Computing connolly surfaces. *J. Mol. Graph.*, **12**, 61–62.
6. Sanner,M.F., Olson,A.J. and Spehner,J.-C. (1995) Fast and robust computation of molecular surfaces. In SCG'95: Proceedings of the eleventh annual symposium on Computational geometry. ACM, New York, NY, USA, pp. 406–407.
7. Varshney,A. and Brooks,F.P. Jr (1993) Fast analytical computation of Richard's smooth molecular surface. In *Proceedings of the IEEE Visualization*. IEEE Computer Society, Washington, DC, USA, pp. 300–307.
8. Cheng,H.-L. and Shi,X. (2005) Quality mesh generation for molecular skin surfaces using restricted union of balls. In *Proceedings of the IEEE Visualization Conference*. IEEE Computer Society, Los Alamitos, CA, USA, p. 51.
9. Cipriano,G. and Gleicher,M. (2007) Molecular surface abstraction. *IEEE Trans. Vis. Comput. Graph.*, **13**, 1608–1615.
10. Goodsell,D. and Olsen,A. (1992) Molecular illustration in black and white. *J. Mol. Graph.*, **10**, 235–240.
11. Jmol: an open-source Java viewer for chemical structures in 3D. http://www.jmol.org/ (7 May 2010, date last accessed).
12. Berman,H.M., Westbrook,J., Feng,Z., Gilliland,G., Bhat,T.N., Weissig,H., Shindyalov,I.N. and Bourne,P.E. (2000) The protein data bank. *Nucleic Acids Res.*, **28**, 235–242.
13. Darnell,S.J., Page,D. and Mitchell,J.C. (2007) An automated decision-tree approach to predicting protein interaction hot spots. *Proteins*, **68**, 813–823.
14. Dolinsky,T.J., Czodrowski,P., Li,H., Nielsen,J.E., Jensen,J.H., Klebe,G. and Baker,N.A. (2007) PDB2PQR: expanding and upgrading automated preparation of biomolecular structures for molecular simulations. *Nucleic Acids Res.*, **35**, W522–W525.

15. Baker,N.A., Sept,D., Joseph,S., Holst,M.J. and McCammon,J.A. (2001) Electrostatics of nanosystems: application to microtubules and the ribosome. *Proc. Natl Acad. Sci. USA*, **98**, 10037–10041.
16. Landis,H. (2002) Production ready global illumination. In *Siggraph Course Notes*.
17. Tarini,M., Cignoni,P. and Montani,C. (2006) Ambient occlusion and edge cueing for enhancing real time molecular visualization. *IEEE Trans. Vis. Comput. Graph.*, **12**, 1237–1244.